AMENDMENTS TO THE SPECIFICATION

Please amend the specification as indicated hereafter. It is believed that the following amendments and additions add no new matter to the present application.

In the Specification: [Use ~~strikethrough~~ for deleted matter (or double square brackets "[[]]" if the strikethrough is not easily perceivable, *i.e.*, "4" or a punctuation mark) and underlined for added matter.]

Please amend the paragraph starting on p. 7, line 6 as follows:

B1

Illustrated in FIG. 3A is a block diagram illustrating the generated program code 40 for the source program code 31 shown in FIG. 2A. First, the generated program code 40 is started at step 41. Next the initialization step is performed at step 42. After performing the initialization, the generated program code 40 then calls generated function code 50 at step 43. Generated function code 50 is herein defined in further detail with regard to FIG. 3B.

Please amend the paragraph starting on p. 9, line 8 as follows:

B2

Illustrated in FIG. 5A is a block diagram of an example source program code 35 that includes a ~~source~~ recovery code 36. The following is a general explanation (by example) describing how the programmer uses pragma to add recovery code to a program. The programmer can put any code needed between the "recover begin" and "recover end" pragmas. This recovery code 36 will be executed if the call to function A returns with the hidden error flag set, otherwise, it is skipped.

Please amend the paragraph starting on p. 9, line 14 as follows:

B3

For instance, the recovery ~~routine~~ code 36 ~~code~~ could be written on the same line as the "#pragma recovery", but that would limit how complex a recovery sequence could be without sacrificing readability.

Please amend the paragraph starting on p. 9, line 23 as follows:

B4

Shown in FIG 5A is an example of a source program code 35. The source program code 35 is first initialized and then performs a call to function A ~~32 (FIG.2B) and~~ 70 (FIG. 4B). After performing the call to function A, the source program code 35 checks if the hidden error flag is set. If the hidden error flag is set, recovery

B4

~~routine~~ code 36 is executed. Otherwise, recovery ~~routine~~ code 36 is skipped and the source program code 35 performs a process on the return data and then determines if processing is done. If processing is done, the source program code 35 then exits.

Please amend the paragraph starting on p. 10, line 6 as follows:

Illustrated in FIG. 5B is an example of a representation of the generated program code 80 for the source program code 35 that includes generated recovery code 100, corresponding to recovery code 36 as shown in FIG. 5A. The generated function code illustrated in FIG. 5B is prepared by the compiler 110 with the software recovery mechanism 120 and includes generated recovery code 100 and the corresponding recovery code 36 shown in FIGs. 2 and 5A, respectively.

B5